A Partial Ranking Algorithm for Resource Allocation Problems

De Waegenaere, A.M.B.; Wielhouwer, J.L.

Publication date:
2001

No. 2001-40

A PARTIAL RANKING ALGORITHM FOR RESOURCE ALLOCATION PROBLEMS

By Anja De Waegenaere and Jacco L. Wielhouwer

June 2001

ISSN 0924-7815

Discussion paper

A Partial Ranking Algorithm for Resource Allocation Problems

Anja De Waegenaere* Jacco L. Wielhouwer†

May 29, 2001

Abstract

We present an algorithm to solve resource allocation problems with a single resource, a convex separable objective function, a convex separable resource-usage constraint and bounded variables. Through evaluation of specific functions in the lower and/or upper bounds, we obtain information on whether or not these bounds are binding. Once this information is available for all variables, the optimum is found through determination of the unique root of a strictly decreasing function. A comparison is made with the currently known most efficient algorithms.

Keywords: Programming, non-linear: resource allocation. Programming, algorithms: partial ranking.

*Corresponding author: CentER for Economic Research and Department of Econometrics and Operations Research, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands. Tel: ++31-13-4662913. Fax: ++31-13-4663280. Email: a.m.b.DeWaegenaere@kub.nl.

†CentER for Economic Research, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands.

1 Introduction

We focus on the continuous variable resource allocation problem of the following form:

$$\begin{aligned} &\min_{(d_1,\ldots,d_N)} \ \sum_{k=1}^{N} f_k(d_k) \\ &s.t. \ \ \sum_{k=1}^{N} g_k(d_k) \leqslant D, \\ &\qquad l_k \leqslant d_k \leqslant u_k, \quad \text{for } k = 1, \ldots, N, \end{aligned} \tag{1}$$

where the performance functions $f_k(.)$ and the resource-usage functions $g_k(.)$ are differentiable and convex, $l_k \in \Re \cup \{-\infty\}$ and $u_k \in \Re \cup \{+\infty\}$. The applied operations research literature provides numerous examples of optimization problems that can be written as a resource allocation problem. For some practical applications see e.g. Helgason et al. (1980), Nielsen and Zenios (1993) and De Waegenaere and Wielhouwer (2001).

Apart from its practical applications, the above problem is also important in heuristics and branch and bound algorithms for solving the integer valued resource allocation problem (see e.g. Bretthauer and Shetty 1995, Ventura and Weng 1995).

For the case of quadratic objective functions, several efficient algorithms have been developed, see e.g. Pardalos and Kovoor (1990) and Shetty and Muthukrishnan (1990). On the solution of problem (1) with convex performance functions and a linear resource-usage constraint, the most efficient methods described until now are due to Zipkin (1980) (which is a generalization and extension of Luss and Gupta 1975), Bitran and Hax (1981), and Nielsen and Zenios (1992). Kodialam and Luss (1998) extend the approaches of Zipkin (1980) and Bitran and Hax (1981) to allow for a more general separable convex resource-usage constraint in case of variables that are unbounded above. Bretthauer and Shetty (1995) focus on the integer valued resource allocation problem and provide a generalization of the approach of Nielsen and Zenios (1992) to solve the real valued subproblems in a branch and bound algorithm.

When comparing these approaches, we see that Bitran and Hax (1981) obtain information on lower or upper bounds that are binding through solving resource allocation problems with unbounded variables, whereas Zipkin (1980) obtains that information through a combination of ranking, specific function evaluations, and solving resource allocation problems with unbounded variables. Once this information is available for all variables, it remains to solve a resource allocation problem with unbounded variables, which is equivalent to solving one equation in one unknown. When the variables are

bounded both below and above, both algorithms in general require several iterations, each of which involves solving an equation. When the variables are unbounded above, Zipkin's algorithm requires only one equation to be solved. However, the variables need to be ranked. Kodialam and Luss (1998) numerically compare these two approaches for variables that are unbounded above and conclude that in cases where there is a closed form solution to the problem with unbounded variables, the algorithm of Bitran and Hax (1981) is more efficient. The opposite holds when no such closed form solution is available. In addition, their numerical examples show that in case of Zipkin's approach, the ranking of the variables on average takes 20%-25% of the total time needed to solve the problem, so that important improvements can be achieved if the ordering time is reduced. Nielsen and Zenios (1992) express all the decision variables as a function of the Lagrange multiplier of the resource-usage constraint, so that only one equation needs to be solved even in the presence of both upper and lower bounds. In order to reduce the numerical complexity of root searching, however, a sequence of length $2N$ is ranked.

We present an algorithm where information on whether or not bounds are binding is obtained exclusively through evaluation of given functions in the lower and/or upper bounds. During these function evaluations, the variables are implicitely ranked partially. Our algorithm therefore does not require the variables to be ranked completely, and a resource allocation problem with unbounded variables has to be solved only once. This is a benefit compared to the above described approaches, since each of these approaches requires full ranking and/or solving multiple equations.

The paper is organized as follows. In section 2 we provide a characterization of the optimal allocation scheme. In section 3 we present an efficient algorithm to calculate it. In section 4, we discuss the computational efficiency of the algorithm, and provide a detailed comparison with the approaches of Zipkin (1980), Bitran and Hax (1981) and Nielsen and Zenios (1992). The paper is concluded in section 5.

2 Characterization of the optimal solution

Given the structure of problem (1), its optimal solution can be found by solving the set of Karush-Kuhn-Tucker (KKT) conditions. The main idea that leads to our algorithm is as follows. Once it is known, for each variable, whether or not a corresponding lower

or upper bound is binding, the optimum can be found by simply determining the unique root of a particular strictly decreasing function. Our algorithm therefore provides an efficient way to determine whether or not lower or upper bounds are binding.
We first introduce the following notation:

- $\mathcal{K}_{lb}$ is the set of variables for which *it is known that* the corresponding lower bound is binding,

- $\mathcal{K}_{ub}$ is the set of variables for which *it is known that* the corresponding upper bound is binding,

- $\mathcal{K}_{lnb}$ is the set of variables for which *it is known that* the corresponding lower bound is not binding,

- $\mathcal{K}_{unb}$ is the set of variables for which *it is known that* the corresponding upper bound is not binding,

- $\mathcal{K}_{bnb}$ is the set of variables for which *it is known that* both bounds are not binding, i.e. $\mathcal{K}_{bnb} = \mathcal{K}_{lnb} \cap \mathcal{K}_{unb}$,

- $\mathcal{K}$ is the set of variables for which *it is known*, for both bounds, whether or not they are binding, i.e. $\mathcal{K} = \mathcal{K}_{lb} \cup \mathcal{K}_{ub} \cup \mathcal{K}_{bnb}$.

In Theorem 1, we show how the optimal solution can be determined given the information that is available. In Theorem 2, we show how additional information can be obtained. We introduce the following notation:

$$F_k(.) = \frac{f_k'(.)}{g_k'(.)}, \quad \text{for all } k = 1, ..., N.$$

We assume that, for all k, $f_k(.)$ and $g_k(.)$ are differentiable and convex, $g_k'(.) \neq 0$, and $F_k(.)$ is continuous and invertible. Moreover, the functions $g_k(F_k^{-1}(.))$ are either all strictly increasing or all strictly decreasing. (Notice that this allows e.g. for $g_k(x) = a_k x$ with $a_k > 0$ for some k, and $a_k < 0$ for the other k's, as in Nielsen and Zenios 1992).

For notational convenience, we will focus on the case where, for all k, it holds that $g_k(.)$ and $F_k(.)$ are strictly increasing and $F_k(.) < 0$. This is satisfied for example when, for all k, $f_k(.)$ is strictly convex and decreasing, and $g_k(.)$ is convex and strictly increasing.

In order to avoid extensive notation to take into account that some of the bounds can be equal to $-\infty$ or $+\infty$, we will implicitly assume the following: for any function $h(.)$, $h(l_k)$ denotes $\lim_{x\to-\infty} h(x)$ in case $l_k = -\infty$. Similarly, $[l_k, u_k]$ denotes $(-\infty, u_k]$ in case $l_k = -\infty$. Similar notations will be used in case $u_k = +\infty$.

First notice that, similarly to Bretthauer and Shetty (1995), one can verify whether or not the inequality constraint is binding in the following way. Consider the following inequality

$$\sum_{k=1}^{N} g_k(u_k) \leqslant D. \tag{2}$$

If (2) is satisfied, the inequality constraint is not binding and the optimum is given by $d_k = u_k$, in case $u_k < +\infty$ for all k. If (2) is satisfied with $u_k = +\infty$ for some k, an optimum does not exist. In the sequel we will therefore always assume that (2) is not satisfied, so that the inequality constraint can be replaced by an equality constraint.

Theorem 1 Take any J, and define the function $\Psi_J(.)$ as follows:

$$\Psi_J(d) := D - g_J(d) - \sum_{\substack{k=1 \\ k\neq J}}^{N} g_k(\xi_k(J,d)), \tag{3}$$

where

$$\begin{aligned}
\xi_k(J,d) &= \min\left\{\max\left\{F_k^{-1}(F_J(d)), l_k\right\}, u_k\right\}, && \text{if } k \notin (\mathcal{K} \cup \mathcal{K}_{lnb} \cup \mathcal{K}_{unb}),\\
&= \min\{F_k^{-1}(F_J(d)), u_k\}, && \text{if } k \in \mathcal{K}_{lnb} \setminus \mathcal{K}_{bnb},\\
&= \max\{F_k^{-1}(F_J(d)), l_k\}, && \text{if } k \in \mathcal{K}_{unb} \setminus \mathcal{K}_{bnb},\\
&= F_k^{-1}(F_J(d)), && \text{if } k \in \mathcal{K}_{bnb},\\
&= l_k, && \text{if } k \in \mathcal{K}_{lb},\\
&= u_k, && \text{if } k \in \mathcal{K}_{ub}.
\end{aligned} \tag{4}$$

Then, the following holds:

i) The function $\Psi_J(.)$ is continuous and strictly decreasing, so that it has at most one root.

ii) Let us denote

$$\mathcal{P} := \left\{ J : \begin{array}{ll} & \Psi_J(u_J) < 0 < \Psi_J(l_J) \\ \text{or} & \Psi_J(l_J) = 0 \text{ and } l_J > -\infty \\ \text{or} & \Psi_J(u_J) = 0 \text{ and } u_J < +\infty \end{array} \right\}$$

The optimum exists iff $\mathcal{P} \neq \emptyset$. The optimal solution satisfies:

$$\begin{cases} d_k = \xi_k(J, d_J), & \text{for all } k \neq J, \\ d_J = \Psi_J^{-1}(0), \end{cases} \tag{5}$$

for any $J \in \mathcal{P}$.

Proof:

i) The fact that $\Psi_J(.)$ is continuous and strictly decreasing follows immediately from the fact that the functions $F_k(.)$ and $g_k(.)$ are continuous and strictly increasing.

ii) Given that (2) is not satisfied and that it is known that for all variables in $\mathcal{K}_{lnb}$ ($\mathcal{K}_{unb}$) the lower (upper) bound is not binding, and that for all variables in $\mathcal{K}_{lb}$ ($\mathcal{K}_{ub}$) the lower (upper) bound is binding, the optimal value for all variables $d_k, k \in \mathcal{S} := \{1, \ldots, N\} \setminus (\mathcal{K}_{lb} \cup \mathcal{K}_{ub})$ can be found by solving the following optimization problem:

$$\begin{array}{ll} \min_{(d_1,\ldots,d_N)} & \sum_{k\in\mathcal{S}} f_k(d_k) \\ s.t. & \sum_{k\in\mathcal{S}} g_k(d_k) = D - \sum_{k\in\mathcal{K}_{lb}} g_k(l_k) - \sum_{k\in\mathcal{K}_{ub}} g_k(u_k), \\ & l_k \leqslant d_k \leqslant u_k, \quad \text{for } k \in \mathcal{S} \setminus (\mathcal{K}_{lnb} \cup \mathcal{K}_{unb}) \\ & d_k \geqslant l_k, \quad \text{for } \; k \in \mathcal{S} \cap (\mathcal{K}_{unb} \setminus \mathcal{K}_{lnb}) \\ & d_k \leqslant u_k, \quad \text{for } \; k \in \mathcal{S} \cap (\mathcal{K}_{lnb} \setminus \mathcal{K}_{unb}) \end{array} \tag{6}$$

with $\{k : l_k = -\infty\} \subset \mathcal{K}_{lnb}$ and $\{k : u_k = +\infty\} \subset \mathcal{K}_{unb}$.

The *Lagrangian* of problem (6) is given by:

$$\begin{aligned} L(d, \lambda, \mu) \;=\; & \sum_{k\in\mathcal{S}} f_k(d_k) - \sum_{k\in\mathcal{S}\setminus\mathcal{K}_{lnb}} \mu_k(d_k - l_k) + \sum_{k\in\mathcal{S}\setminus\mathcal{K}_{unb}} \tau_k(d_k - u_k) \\ & + \lambda \left(\sum_{k\in\mathcal{S}} g_k(d_k) + \sum_{k\in\mathcal{K}_{lb}} g_k(l_k) + \sum_{k\in\mathcal{K}_{ub}} g_k(u_k) - D \right). \end{aligned} \tag{7}$$

Given that $g_k'(.) > 0$, the necessary and sufficient conditions for an optimum are as

follows: there exist $\mu_k \geqslant 0$, $\tau_k \geqslant 0$, and $\lambda \geqslant 0$, such that

$$\begin{cases} F_k(d_k) + \lambda = 0, & k \in \mathcal{K}_{bnb}, \\ F_k(d_k) + \lambda - \mu_k = 0, & k \in \mathcal{S} \cap (\mathcal{K}_{unb} \setminus \mathcal{K}_{lnb}), \\ F_k(d_k) + \lambda + \tau_k = 0, & k \in \mathcal{S} \cap (\mathcal{K}_{lnb} \setminus \mathcal{K}_{unb}), \\ F_k(d_k) + \lambda - \mu_k + \tau_k = 0, & k \in \mathcal{S} \setminus (\mathcal{K}_{lnb} \cup \mathcal{K}_{unb}), \\ \mu_k(d_k - l_k) = 0, & k \in \mathcal{S} \setminus \mathcal{K}_{lnb}, \\ \tau_k(d_k - u_k) = 0, & k \in \mathcal{S} \setminus \mathcal{K}_{unb}, \\ \sum_{k=1}^{N} g_k(d_k) = D, & \\ l_k \leqslant d_k \leqslant u_k, & k \in \mathcal{S} \setminus (\mathcal{K}_{lnb} \cup \mathcal{K}_{unb}), \\ d_k \geqslant l_k, & k \in \mathcal{S} \cap (\mathcal{K}_{unb} \setminus \mathcal{K}_{lnb}), \\ d_k \leqslant u_k, & k \in \mathcal{S} \cap (\mathcal{K}_{lnb} \setminus \mathcal{K}_{unb}). \end{cases} \tag{8}$$

Let $(d_1, ..., d_N)$ be the unique allocation scheme that satisfies (5) for a given $J \in \mathcal{P}$. We will show that there exist μ_k, τ_k, and λ such that the conditions in (8) are satisfied. First notice that $i)$ and $J \in \mathcal{P}$ imply that $d_J := \Psi_J^{-1}(0) \in [l_J, u_J]$. Moreover, $\Psi_J(d_J) = 0$ implies that $\sum_{k=1}^{N} g_k(d_k) = D$.
Finally, it is seen immediately that with

$$\begin{aligned} \mu_k &:= \max\{F_k(l_k), F_J(d_J)\} - F_J(d_J) \geqslant 0, && k \in \mathcal{S} \setminus \mathcal{K}_{lnb}, \\ \tau_k &:= F_J(d_J) - \min\{F_k(u_k), F_J(d_J)\} \geqslant 0, && k \in \mathcal{S} \setminus \mathcal{K}_{unb}, \\ \lambda &:= -F_J(d_J) \geqslant 0, \end{aligned} \tag{9}$$

all the necessary and sufficient conditions for optimality are satisfied.
Now suppose that d_k, μ_k, τ_k and λ satisfy conditions (8). It is clear that $\mathcal{K}_{bnb} \cup \{k : \mu_k = \tau_k = 0\} \subset \mathcal{P}$. Indeed, take $J \in \mathcal{K}_{bnb} \cup \{k : \tau_k = \mu_k = 0\}$ and set $d_J := F_J^{-1}(-\lambda)$. Then (8) implies that $d_k = \xi_k(J, d_J)$ and $\sum_{k=1}^{N} g_k(d_k) = D$ implies that $\Psi_J(d_J) = 0$, so that $J \in \mathcal{P}$. Now suppose that $\mathcal{K}_{bnb} \cup \{k : \mu_k = \tau_k = 0\} = \emptyset$. Then, if there exists a $\tau_k > 0$, let J be such that $\tau_J = \min\{\tau_k | k : \tau_k > 0\}$. Otherwise, let J be such that $\mu_J = \min\{\mu_k | k : \mu_k > 0\}$. It can then be verified that $J \in \mathcal{P}$. We can therefore conclude that $\mathcal{P} \neq \emptyset$, and that $(d_1, ..., d_N)$ satisfies (5) for any $J \in \mathcal{P}$.
This concludes the proof. □

The above theorem implies that the optimal solution can be found by determining the root of $\Psi_J(.)$ for any $J \in \mathcal{P}$. It is clear that in case $\Psi_J(u_J) < 0 < \Psi_J(l_J)$, the root searching procedure will become more efficient as more information becomes available

on whether or not bounds are binding, i.e. $\mathcal{K}$ becomes larger. In order to increase the set $\mathcal{K}$, we will use the following fundamental result, where the optimal solution is denoted $(d_1^*, \ldots, d_N^*)$.

Theorem 2 For any $J \in \{1, \ldots, N\}$, the following holds:

i) If $\Psi_J(l_J) < 0$, then $\begin{cases} d_k^* = l_k, \text{ for all } k \text{ for which } F_k(l_k) \geqslant F_J(l_J), \\ d_k^* < u_k, \text{ for all } k \text{ for which } F_k(u_k) \geqslant F_J(l_J). \end{cases}$

ii) If $\Psi_J(l_J) > 0$, then $\begin{cases} d_k^* > l_k, \text{ for all } k \text{ for which } F_k(l_k) \leqslant F_J(l_J), \\ d_k^* = u_k, \text{ for all } k \text{ for which } F_k(u_k) \leqslant F_J(l_J). \end{cases}$

iii) If $\Psi_J(u_J) > 0$, then $\begin{cases} d_k^* = u_k, \text{ for all } k \text{ for which } F_k(u_k) \leqslant F_J(u_J), \\ d_k^* > l_k, \text{ for all } k \text{ for which } F_k(l_k) \leqslant F_J(u_J). \end{cases}$

iv) If $\Psi_J(u_J) < 0$, then $\begin{cases} d_k^* < u_k, \text{ for all } k \text{ for which } F_k(u_k) \geqslant F_J(u_J), \\ d_k^* = l_k, \text{ for all } k \text{ for which } F_k(l_k) \geqslant F_J(u_J). \end{cases}$

v) If $\Psi_J(l_J) = 0$ $(\Psi_J(u_J) = 0)$, then $d_k^* = \xi_k(J, l_J)$ $(d_k^* = \xi_k(J, u_J))$ for all k.

Proof: *i*) Suppose that $\mu_J = 0$ in the optimal solution i.e. the lower bound for variable J is not binding. Then there exists a $\tau_J \geqslant 0$ such that the conditions in (8) are satisfied with

$$\lambda = -F_J(d_J^*) - \tau_J. \tag{10}$$

Now let us denote $\widetilde{\Psi}_J(\cdot)$ for the function that equals $\Psi_J(\cdot)$ but with $F_J(d)$ replaced by $F_J(d) + \tau_J$. It can now be seen from the proof of Theorem 1 that the optimum satisfies $\widetilde{\Psi}_J(d_J^*) = 0$. Since $\widetilde{\Psi}_J(\cdot)$ is clearly decreasing and $d_J^* \geqslant l_J$, it follows that $\widetilde{\Psi}_J(l_J) \geqslant 0$. Now $\tau_J \geqslant 0$ implies that $\Psi_J(l_J) \geqslant \widetilde{\Psi}_J(l_J) \geqslant 0$.
Therefore, by contradiction, $\Psi_J(l_J) < 0$ implies that $\mu_J > 0$, which implies $d_J^* = l_J$. Combined with conditions (8), this yields the proof of *i*).
ii) Using similar arguments as in the proof of *i*) one can show that $\Psi_J(l_J) > 0$ implies that $\mu_J = 0$. Now since $\mu_J = 0$ and $d_J^* = l_J$ implies that $\Psi_J(l_J) = 0$, we can conclude that $d_J^* > l_J$. Combined with conditions (8), this yields the proof of *ii*).
iii), *iv*) The proof is similar.

v) Follows immediately from *ii*) in Theorem 1.
This concludes the proof. □

The above theorem states that, with each evaluation of $\Psi_J(l_J)$ or $\Psi_J(u_J)$ for some J, additional information can be obtained on whether or not certain bounds are binding. This information is extremely useful since it follows immediately from Theorem 1 that when it is known that the lower bound for the k^{th} variable is binding (resp. not binding), then the term $\max\{l_k\ ,\ F_k^{-1}(F_J(d))\}$ in $\Psi_J(.)$ can be replaced by its first (resp. second) argument. A similar argument holds for the upper bounds. This clearly simplifies the procedure in any following step.

3 The Algorithm

Theorem 2 implies that evaluation of $\Psi_J(l_J)$ and/or $\Psi_J(u_J)$ for different values of J yields information on whether or not bounds are binding, i.e. it increases the set $\mathcal{K}$. In particular, one can infer from Theorem 2 that

$$\begin{aligned}\mathcal{K}_{bnb} &\supset \{J : \Psi_J(u_J) < 0 < \Psi_J(l_J)\},\\ \mathcal{K}_{lb} &\supset \{J : \Psi_J(l_J) < 0\},\\ \mathcal{K}_{ub} &\supset \{J : \Psi_J(u_J) > 0\}.\end{aligned}$$

Notice that Theorem 2 implies that evaluation of a $\Psi_J(l_J)$ or $\Psi_J(u_J)$ can yield information on whether or not bounds are binding for several other variables. For example, if $\Psi_J(l_J) < 0$ so that $J \in \mathcal{K}_{lb}$, then $k \in \mathcal{K}_{lb}$ for all k with $F_k(l_k) \geqslant F_J(l_J)$.
Theorem 1 implies that, as soon as at least one element $J \in \mathcal{K}_{bnb} \cup \left\{ J : \begin{smallmatrix} \Psi_J(l_J)=0 \text{ and } l_J>-\infty \\ \text{or } \Psi_J(u_J)=0 \text{ and } u_J<+\infty \end{smallmatrix} \right\}$ is known, the optimum can be found by determining the root of the function $\Psi_J(.)$. One can therefore distinguish the following two extreme approaches for solving the resource allocation problem:

1. Evaluation of $\Psi_J(l_J)$ and/or $\Psi_J(u_J)$ is continued until $\Psi_J(l_J) = 0$ or $\Psi_J(u_J) = 0$ or $\mathcal{K} = \{1, \ldots, N\}$. In the latter case the root of $\Psi_J(.)$ is determined for a $J \in \mathcal{K}_{bnb}$.

2. Root searching is started as soon as an element in $\mathcal{K}_{bnb}$ is found (unless $\Psi_J(l_J) = 0$ or $\Psi_J(u_J) = 0$ before that).

It is intuitively clear that approach 2 on average would be computationally less efficient, since the function $\Psi_J(.)$ in general will still contain a number of maximum and minimum

terms when its root has to be determined. This is not the case for approach 1. It can be shown that the efficiency of approach 2 can be improved through a result that allows to combine root searching with additional information gathering. However, the efficiency will then largely depend on the choice of the root searching algorithm. Therefore we present the algorithm based on approach 1. Remember that $\mathcal{K} = \mathcal{K}_{lb} \cup \mathcal{K}_{ub} \cup \mathcal{K}_{bnb}$ and $\mathcal{S} = \{1, \ldots, N\} \setminus (\mathcal{K}_{lb} \cup \mathcal{K}_{ub})$.

Algorithm DW-W

Step 0: Set $\mathcal{K}_{lb} = \mathcal{K}_{ub} = \emptyset$, and $\mathcal{K}_{lnb} = \{k : l_k = -\infty\}$, $\mathcal{K}_{unb} = \{k : u_k = +\infty\}$.

Step 1: Pick any $J \notin \mathcal{K}$. If $J \in \mathcal{K}_{lnb}$, go to Step 3.

Step 2: If $\Psi_J(l_J) = 0$: STOP the optimum is $d_k^* = \xi_k(J, l_J)$.
If $\Psi_J(l_J) < 0$ then:

- $\mathcal{K}_{lb} = \mathcal{K}_{lb} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{lnb} : F_k(l_k) \geqslant F_J(l_J)\}$
- $\mathcal{K}_{unb} = \mathcal{K}_{unb} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{unb} : F_k(u_k) \geqslant F_J(l_J)\}$
- Go to Step 4.

If $\Psi_J(l_J) > 0$, then:

- $\mathcal{K}_{lnb} = \mathcal{K}_{lnb} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{lnb} : F_k(l_k) \leqslant F_J(l_J)\}$
- $\mathcal{K}_{ub} = \mathcal{K}_{ub} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{unb} : F_k(u_k) \leqslant F_J(l_J)\}$
- If $J \in \mathcal{K}_{unb}$, go to Step 4.

Step 3: If $\Psi_J(u_J) = 0$: STOP the optimum is $d_k^* = \xi_k(J, u_J)$.
If $\Psi_J(u_J) > 0$ then:

- $\mathcal{K}_{ub} = \mathcal{K}_{ub} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{unb} : F_k(u_k) \leqslant F_J(u_J)\}$
- $\mathcal{K}_{lnb} = \mathcal{K}_{lnb} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{lnb} : F_k(l_k) \leqslant F_J(u_J)\}$

If $\Psi_J(u_J) < 0$ then:

- $\mathcal{K}_{unb} = \mathcal{K}_{unb} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{unb} : F_k(u_k) \geqslant F_J(u_J)\}$

– $\mathcal{K}_{lb} = \mathcal{K}_{lb} \cup \{k \in \mathcal{S} \setminus \mathcal{K}_{lnb} : F_k(l_k) \geqslant F_J(u_J)\}$

Step 4: If $\mathcal{K} \neq \{1, \ldots, N\}$, go to Step 1.
Else, go to Step 5.

Step 5: Determine the root $d_J^* \in [l_J, u_J]$ of $\Psi_J(.)$ for any $J \in \mathcal{K}_{bnb}$, and, $d_k^* = \xi_k(J, d_J^*)$.

Theorem 3 Algoritm DW-W stops after a finite number of iterations, and yields the optimum if it exists.

Proof: It is clear that either Step 5 of the algorithm is reached, or there is a J such that $\Psi_J(l_J) = 0$ or $\Psi_J(u_J) = 0$. The latter implies that the optimum is found before Step 5 is reached and the algorithm indeed stops. Notice that when the optimum is such that each of the variables is either at its upper or at its lower bound, then it will be the case that Step 5 is not reached. Indeed, it can be seen from the proof of Theorem 1 that, at the latest when $\mathcal{K}$ contains all but one elements, it will be the case that $\Psi_J(.)$ equals zero either in an upper or in a lower bound. Therefore, when Step 5 is reached, an optimum exists iff $\mathcal{P} = \mathcal{K}_{bnb} \neq \emptyset$ and $\Psi_J(.)$ has a root in $[l_J, u_J]$ for all $J \in \mathcal{K}_{bnb}$. It follows from Theorem 1 that root searching in Step 5 for *one arbitrary* $J \in \mathcal{K}_{bnb}$ yields either the optimal solution or the knowledge that an optimal solution does not exist. □

4 Computational Efficiency

In this section we first determine the computational complexity of our algorithm. Then we compare it to the algorithms of Zipkin (1980), Bitran and Hax (1981) and Nielsen and Zenios (1992).
The computations involved in algorithm DW-W are as follows.

- A number of evaluations of $\Psi_J(l_J)$, each of which requires a number of comparisons between $F_J(l_J)$ and $F_k(l_k)$, and/or $F_k(u_k)$.

- A number of evaluations of $\Psi_J(u_J)$, each of which requires a number of comparisons between $F_J(u_J)$ and $F_k(u_k)$, and/or $F_k(l_k)$.

- Solving *one* equation of the form $\Psi_J(d) = 0$, where

$$\Psi_J(d) = D - \sum_{\substack{k \in \mathcal{K}_{bnb} \\ k \neq J}} g_k(F_k^{-1}(F_J(d))) - g_J(d) - \sum_{k \in \mathcal{K}_{lb}} g_k(l_k) - \sum_{k \in \mathcal{K}_{ub}} g_k(u_k). \tag{11}$$

Regarding the efficiency of root searching, notice that when root searching is started, it holds that $\mathcal{K} = \{1, ..., N\}$, so that $\mathcal{K}_{lb}(\mathcal{K}_{ub})$, contains all the variables for which the lower bound is binding. Therefore, $F_k^{-1}(.)$ only has to be determined for those variables for which both bounds are unbinding. Moreover, a particular choice of $J \in \mathcal{K}_{bnb}$ implies that:

- The root of $\Psi_J(.)$ is in $[l_J, u_J]$ if it exists.
- $F_J^{-1}(d)$ need not be evaluated.

Therefore, efficiency can be gained through careful choice of J.

The following proposition determines the order of complexity of the function evaluations and the comparisons.

Proposition 1 When $J \notin \mathcal{K}$ is chosen randomly in Step 1 of algorithm DW-W, the following holds:

i) The average number of evaluations of $\Psi_J(l_J)$ and of $\Psi_J(u_J)$ is $\mathcal{O}(\ln N)$. More precisely the average is less than $4 \ln N$.

ii) The average number of minima (maxima) to be determined is $\mathcal{O}(N)$. More precisely the average is less than $4N$.

Proof: Let $\mathcal{U}_l(n)$ denote the set of variables for which it is *unknown* whether or not the corresponding lower bound is binding after n evaluations of $\Psi_J(l_J)$, and let $\mathcal{U}_{lb}(n)$ (resp. $\mathcal{U}_{lnb}(n)$) denote the subset of variables for which the lower bound *is* binding (resp. not binding).
Finally, denote $I_l(n+1)$ for the random variable that yields the number of variables for which evaluation of the $n+1^{th}$ $\Psi_J(l_J)$ provides information on whether or not the corresponding lower bound is binding.

We first show that, in expectation, the number of maximum terms reduces with at least 25%, and at most 50% with each evaluation of $\Psi_J(l_J)$, i.e.

$$\frac{U_l(n)}{4} \leqslant E[I_l(n+1)] \leqslant \frac{U_l(n)}{2}. \tag{12}$$

where $U_l(n) := \#\mathcal{U}_l(n)$.

It is clear that with $J \in \mathcal{U}_l(n)$ randomly chosen, the following holds:

$$\begin{aligned} P(J \in \mathcal{U}_{lb}(n)) &= \frac{U_{lb}(n)}{U_{lb}(n)+U_{lnb}(n)}, \\ P(J \in \mathcal{U}_{lnb}(n)) &= \frac{U_{lnb}(n)}{U_{lb}(n)+U_{lnb}(n)}, \end{aligned}$$

where $U_{lb}(n) := \#\mathcal{U}_{lb}(n)$, and $U_{lnb}(n) := \#\mathcal{U}_{lnb}(n)$.

It then follows from Theorem 2 *i*) and *ii*), and the fact that J is randomly chosen, that:

$$\begin{aligned} E[I_l(n+1)|J \in \mathcal{U}_{lb}(n)] &= U_{lb}(n)/2 \\ E[I_l(n+1)|J \in \mathcal{U}_{lnb}(n)] &= U_{lnb}(n)/2. \end{aligned}$$

This yields:

$$\begin{aligned} E[I_l(n+1)] &= E[I_l(n+1)|J \in \mathcal{U}_b(n)]P(J \in \mathcal{U}_b(n)) \\ &\quad + E[I_l(n+1)|J \in \mathcal{U}_{nb}(n)]P(J \in \mathcal{U}_{nb}(n)) \\ &= \tfrac{U_{lb}^2(n)+U_{lnb}^2(n)}{2(U_{lb}(n)+U_{lnb}(n))} \end{aligned}$$

Since $U_{lb}(n) + U_{lnb}(n) = U_l(n)$, this implies that

$$\frac{U_l(n)}{4} \leqslant E[I_l(n+1)] \leqslant \frac{U_l(n)}{2}. \tag{13}$$

Now since

$$U_l(n+1) = U_l(n) - I_l(n+1), \tag{14}$$

(13) implies that

$$E[U_l(n+1)] \leqslant \frac{3}{4}E[U_l(n)] \leqslant \left(\frac{3}{4}\right)^{n+1} U_l(0) = \left(\frac{3}{4}\right)^{n+1} N. \tag{15}$$

It is clear that the above argument can be repeated for the upper bounds, so that

$$E[U_u(n+1)] \leqslant \left(\frac{3}{4}\right)^{n+1} N, \tag{16}$$

where $U_u(n+1)$ denotes the number of variables for which it is *unknown* whether or not the corresponding upper bound is binding after $n+1$ evaluations of $\Psi_J(u_J)$.

i) It follows from (15) and (16) that the expected total number of evaluations of $\Psi_J(l_J)$ and of $\Psi_J(u_J)$ needed to eliminate all maximum and minimum terms is less than or equal to the minimal n that satisfies:

$$\left(\frac{3}{4}\right)^n N \leqslant 1. \tag{17}$$

It is clear that (17) is satisfied for

$$n = \ln(N)/\ln\left(\frac{4}{3}\right) < 4\ln N. \tag{18}$$

ii) (15) and (16) imply that the expected total number of maxima (minima) to be determined in all evaluations of $\Psi_J(l_J)$ is less than

$$\sum_{n=0}^{\infty}\left(\frac{3}{4}\right)^n N = 4N. \tag{19}$$

Similarly, the expected total number of maxima (minima) to be determined in all evaluations of $\Psi_J(u_J)$ is less than $4N$. This concludes the proof. □

In the worst case, the number of evaluations of $\Psi_J(\cdot)$ in a lower or upper bound is $\mathcal{O}(N)$, and the number of minima/maxima to be computed is $\mathcal{O}(N^2)$.

In the following three sections we compare algorithm DW-W with those of Zipkin (1980), Bitran and Hax (1981) and Nielsen and Zenios (1992).

4.1 Comparison with Zipkin (Z)

For a specific set of objective functions and a linear equality constraint, Luss and Gupta (1975) derived an algorithm that starts by solving optimization problem (1) without the upper bounds. Then, all the variables that exceed their upper bound are fixed at their upper bound, and the procedure is repeated with a smaller problem until no upper bounds are violated. Each iteration therefore requires solving a problem with only lower bounds. In order to solve these problems with only lower bounds, the variables are ranked such that $F_k(l_k) \leqslant F_{k+1}(l_{k+1})$ for all $k \leqslant N-1$. This ranking implies that in the optimal solution there exists a J such that the lower bounds for $k = 1, \ldots, J$ are not

binding, and the lower bounds for $k = J+1, \ldots, N$ are binding. This J is found through solving resource allocation problems with d_k unbounded for $k \leqslant J$, and $d_k = l_k$ for $k > J$. The objective functions considered by Luss and Gupta (1975) are such that the problem with unbounded variables has a closed form solution. Zipkin (1980) generalized and extended the algorithm to the more general case of performance functions $f_k(.)$ that are differentiable and strictly convex. Moreover, J is obtained through function evaluations rather than through solving problems with unbounded variables. Kodialam and Luss (1998) extend this approach to allow for a separable convex resource-usage function in the case where the variables are unbounded above. A generalization of this approach to allow for upper bounds leads to the following algorithm.

Algorithm Z

Step 0: Set $\mathcal{K}_{ub} = \emptyset$, and rank the variables such that:

$$F_k(l_k) \leqslant F_{k+1}(l_{k+1}), \qquad \text{for all } k \leqslant N-1. \tag{20}$$

Step 1: Find $J^* := \max\{J \notin \mathcal{K}_{ub} : \widetilde{\Psi}_J(-F_J(l_J)) \geqslant 0\}$, where

$$\widetilde{\Psi}_J(\lambda) := D - \sum_{\substack{k=1 \\ k \notin \mathcal{K}_{ub}}}^{J} g_k\big(F_k^{-1}(-\lambda)\big) - \sum_{\substack{k=J+1 \\ k \notin \mathcal{K}_{ub}}}^{N} g_k(l_k) - \sum_{k \in \mathcal{K}_{ub}} g_k(u_k). \tag{21}$$

Step 2: Determine the unique root λ^* of $\widetilde{\Psi}_{J^*}(.)$, and set

$$\begin{aligned} d_k^* &= F_k^{-1}(-\lambda^*), \qquad \text{for } k = 1, \ldots, J^*, \quad k \notin \mathcal{K}_{ub} \\ d_k^* &= l_k, \qquad \text{for } k = J^*+1, \ldots, N\ , \quad k \notin \mathcal{K}_{ub}. \end{aligned}$$

Step 3: If $d_k^* \leqslant u_k$, for all k, then STOP, else, set $\mathcal{K}_{ub} := \mathcal{K}_{ub} \cup \{k : d_k^* > u_k\}$, and go to Step 1.

As is shown in Bitran and Hax (1981), the above algorithm in general requires several iterations, each of which involves finding the root of a function $\widetilde{\Psi}_{J^*}(.)$. Algorithm DW-W only requires one equation to be solved. Moreover, regarding root searching:

- In all iterations in algorithm Z, equation $\widetilde{\Psi}_{J^*}(\lambda) = 0$ has to be solved for $\lambda \in \Re^+$, whereas equation $\Psi_J(d) = 0$ has to be solved only once for $d \in [l_J, u_J]$.

- In the last iteration in algorithm Z, the function $\widetilde{\Psi}_{J^*}(.)$ is identical to the function $\Psi_J(.)$ in Step 5 of algorithm DW-W, except for the fact that the former requires evaluation of $F_J^{-1}(-\lambda)$ whereas the latter requires evaluation of $F_J(d)$. This is a benefit of algorithm DW-W in cases where the inverse has to be determined numerically.

Algorithm Z on average requires $\mathcal{O}(\ln N)$ different evaluations of $\widetilde{\Psi}_J(-F_J(l_J))$ per iteration step. Indeed the most efficient algorithms for determining J^* in Step 1 of algorithm Z are $\mathcal{O}(\ln N)$. As can be seen from Proposition 1, algorithm DW-W on average requires $\mathcal{O}(\ln N)$ different evaluations of $\Psi_J(l_J)$ and of $\Psi_J(u_J)$. Finally, a disadvantage of algorithm Z is that the variables have to be ranked first. In algorithm DW-W, the variables are implicitely ranked partially during the evaluation of $\Psi_J(l_J)$ and $\Psi_J(u_J)$. As is shown in Proposition 1, the average number of maxima and minima to be determined in algorithm DW-W is $\mathcal{O}(N)$, whereas the most efficient ranking algorithms are $\mathcal{O}(N \ln N)$ (see e.g. Harel 1989).

The following table summarizes the average number of computations involved in algorithm Z and in algorithm DW-W, where i denotes the number of iterations required in algorithm Z. The first row gives the average number of maxima and minima that must be determined, and the second row presents the average number of evaluations of $\Psi_J(\cdot)$ ($\widetilde{\Psi}_J(\cdot)$) before root searching starts. The last row indicates how often a root has to be found.

	DW-W	Z
# max(min)	$\mathcal{O}(N)$	$\mathcal{O}(N \ln N)$
$\#\Psi(\cdot), \widetilde{\Psi}(\cdot)$	$\mathcal{O}(\ln N)$	$i * \mathcal{O}(\ln N)$
# roots	1	$N \geqslant i \geqslant 1$

In the case where the variables are unbounded above, it is clear that algorithm Z only requires one iteration, so that the above comparison holds for $i = 1$. Algorithm DW-W in that case reduces to:

Step 0: Set $\mathcal{K}_{lb} = \mathcal{K}_{lnb} = \emptyset$.

Step 1: Pick any $J \notin (\mathcal{K}_{lb} \cup \mathcal{K}_{lnb})$.

Step 2: If $\Psi_J(l_J) = 0$: STOP the optimum is found.
If $\Psi_J(l_J) < 0$ then: $\mathcal{K}_{lb} = \mathcal{K}_{lb} \cup \{k : F_k(l_k) \geqslant F_J(l_J)\}$
If $\Psi_J(l_J) > 0$, then: $\mathcal{K}_{lnb} = \mathcal{K}_{lnb} \cup \{k : F_k(l_k) \leqslant F_J(l_J)\}$

Step 3: If $\mathcal{K}_{lb} \cup \mathcal{K}_{lnb} \neq \{1, \ldots, N\}$, go to Step 1.
Else, go to Step 4.

Step 4: Determine the root of $\Psi_J(.)$ for any $J \in \mathcal{K}_{lnb}$.

Since Zipkin's approach requires full ranking and computationally more intensive root searching, we see that efficiency is gained also in this case.

4.2 Comparison with Bitran and Hax (B-H)

The main idea of the algorithm of Bitran and Hax (1981) is to gather information on whether or not certain lower or upper bounds are binding by solving optimization problem (1) without the lower and upper bound constraints. If the total excess with respect to the upper bounds is higher (lower) than the total shortage with respect to the lower bounds, the all variables that exceed their upper bound (are lower than their lower bound) are fixed at their upper (lower) bound. Subsequently, D is reduced with the appropriate amount and the procedure is repeated with a smaller unconstrained problem until no bounds are violated. Kodialam and Luss (1998) extend this approach to allow for a convex resource-usage function in the case where the variables are unbounded above. A generalization of this approach to allow for upper bounds leads to the following algorithm.

Algorithm B-H

Step 0: Set $\mathcal{K}_{lb} = \mathcal{K}_{ub} = \emptyset$.

Step 1: Determine the unique root λ^* of $\overline{\Psi}(.)$, where

$$\overline{\Psi}(\lambda) := D - \sum_{k \notin (\mathcal{K}_{lb} \cup \mathcal{K}_{ub})} g_k(F_k^{-1}(-\lambda)) - \sum_{k \in \mathcal{K}_{lb}} g_k(l_k) - \sum_{k \in \mathcal{K}_{ub}} g_k(u_k). \tag{22}$$

Step 2: Set $\mathcal{T}_l := \sum_{k \notin (\mathcal{K}_{lb} \cup \mathcal{K}_{ub})} \max\{l_k - F_k^{-1}(-\lambda^*), 0\}$, and
$\mathcal{T}_u := \sum_{k \notin (\mathcal{K}_{lb} \cup \mathcal{K}_{ub})} \max\{F_k^{-1}(-\lambda^*) - u_k, 0\}$.

Step 3: If $\mathcal{T}_l = \mathcal{T}_u = 0$, then STOP.
If $\mathcal{T}_l \geqslant \mathcal{T}_u$, then $\mathcal{K}_{lb} := \mathcal{K}_{lb} \cup \{k : F_k^{-1}(-\lambda^*) < l_k\}$.
If $\mathcal{T}_l \leqslant \mathcal{T}_u$, then $\mathcal{K}_{ub} := \mathcal{K}_{ub} \cup \{k : F_k^{-1}(-\lambda^*) > u_k\}$.
If $\mathcal{T}_l > 0$ or $\mathcal{T}_u > 0$, go to Step 1.

In comparison with algorithm DW-W, we see that algorithm B-H finds information on lower or upper bounds that *are binding* by solving an equation of the form $\overline{\Psi}(\lambda) = 0$, whereas algorithm DW-W finds information on bounds that *can either be binding or not binding* by evaluating $\Psi_J(l_J)$ and/or $\Psi_J(u_J)$ for some values of J. The benefit of algorithm DW-W therefore is that a root has to be determined only once, whereas in general this has to be done multiple times in algorithm B-H. Moreover,

- In all iterations, the equation $\overline{\Psi}(\lambda) = 0$ has to be solved for $\lambda \in \Re^+$.
- In all but the last iteration in algorithm B-H it holds that $\mathcal{K}_{ub}(\mathcal{K}_{lb})$ does not yet contain all the variables for which the upper bound (lower bound) is binding. Therefore, in all iterations in algorithm B-H, the number of terms of the form $F_k^{-1}(.)$ in (22) is strictly larger than in (11) in algorithm DW-W.
- When at least one bound is binding, at least two iterations are needed.

The above makes clear that algorithm DW-W is more efficient than algorithm B-H in cases where there is no closed form solution to the equation $\overline{\Psi}(\lambda) = 0$.

4.3 Comparison with Nielsen and Zenios (N-Z)

In the algorithm of Nielsen and Zenios (1992), the optimal values are expressed as a function of the Lagrange multiplier λ of the constraint $\sum_{k=1}^{N} a_k d_k = D$. Then the optimum is found by solving an equation in λ. Bretthauer and Shetty (1995) extend this approach to allow for a convex resource-usage function and use it in a branch and bound algorithm to solve the integer valued problem. In order to solve the equation in Nielsen and Zenios (1992), the set $\{F_k(l_k), F_k(u_k) : k = 1, \ldots, N\}$ is ranked.

This leads to the following algorithm.

Algorithm N-Z

Define the function

$$\widehat{\Psi}(\lambda) := D - \sum_{k=1}^{N} g_k(\max\{\min\{F_k^{-1}(-\lambda), u_k\}, l_k\}). \tag{23}$$

Step 1: Rank the set $\{-F_k(l_k), -F_k(u_k) : k = 1, \ldots, N\}$ in decreasing order, and denote x_i for the i^{th} element in this set.

Step 2: Find i such that $\widehat{\Psi}(x_i) \geqslant 0$ and $\widehat{\Psi}(x_{i+1}) \leqslant 0$.

Step 3: Find λ^* in $[x_i, x_{i+1}]$ such that $\widehat{\Psi}(\lambda^*) = 0$. The optimum is given by $d_k^* = \max\{\min\{F_k^{-1}(-\lambda^*), u_k\}, l_k\}$.

The following table summarizes the average number of computations involved in algorithm N-Z and algorithm DW-W. The first row gives the average number of maxima and minima that must be determined, and the second row presents the average number of evaluations of $\Psi_J(\cdot)$ ($\widehat{\Psi}(\cdot)$) before root searching starts. The last row indicates how often a root has to be found.

	DW-W	N-Z
# max(min)	$\mathcal{O}(N)$	$\mathcal{O}(N \ln N)$
$\#\Psi(\cdot), \widehat{\Psi}(\cdot)$	$\mathcal{O}(\ln N)$	$\mathcal{O}(\ln N)$
# roots	1	1

We see that both approaches require only one equation to be solved. However, in algorithm N-Z

- A set of $2N$ elements has to be ranked. The most efficient ranking algorithms for N elements are on average $\mathcal{O}(N \ln N)$. In algorithm DW-W the average number of maxima/minima to be computed is only $\mathcal{O}(N)$ due to partial ranking.

- The number of terms of the form $F_k^{-1}(.)$ in (23) is strictly larger than in (11) in algorithm DW-W. Furthermore, efficiency in root searching in algorithm DW-W can be positively affected by choosing a particular $J \in \mathcal{K}_{bnb}$, since this implies that $F_J^{-1}(.)$ need not be evaluated.

5 Conclusion

In this paper we present a new algorithm for solving resource allocation problems with bounded variables, a separable convex objective function and a separable convex resource-usage constraint. We provide a thorough comparison with the most efficient existing approaches which are due to Zipkin (1980), Bitran and Hax (1981) and Nielsen and Zenios (1992). We reformulate these algorithms in order to make them mutually comparable.

These three algorithms require complete ranking of a sequence of minimal length N and/or several iterations in which a root of a monotone function has to be determined. More precisely, the algorithms of Zipkin (1980) and Bitran and Hax (1981) require several roots to be found. In addition to that, the algorithm of Zipkin (1980) requires complete ranking of a sequence of length N. The algorithm of Nielsen and Zenios (1992) requires only one root to be found, but complete ranking of a sequence of length $2N$.

In contrast, our algorithm requires partial ranking instead of full ranking, which reduces the complexity of ordering from $\mathcal{O}(N \ln N)$ to $\mathcal{O}(N)$. A root has to be determined only once. In addition, the function for which the root has to be found is computationally less complex.

Finally, as a side effect, our paper contributes to the literature in the sense that algorithms for the approaches of Zipkin (1980) and Bitran and Hax (1981) are formulated for problems with a convex resource-usage constraint and variables that are bounded from below and above.

References

[1] Bitran G.R., and A.C. Hax (1981), Disaggregation and Resource Allocation Using Convex Knapsack Problems with Bounded Variables, *Management Science* 27, 431-441.

[2] Bretthauer K.M., and B. Shetty (1995), The Nonlinear Resource Allocation Problem, *Operations Research* 43, 670-683.

[3] De Waegenaere A., and J.L. Wielhouwer (2001), Optimal Tax Depreciation Lives and Charges under Regulatory Constraints, *CentER Discussion Paper 2001-23*.

[4] Harel D. (1989) *The Science of Computing: exploring the nature and power of algorithms*, Addison-Wesley, Reading, MA.

[5] Helgason R., J. Kennington, and H. Lall (1980), A Polynomial Bounded Algorithm for a Single Constrained Quadratic Program, *Mathematical Programming* 18, 338-343.

[6] Kodialam M.S., and H. Luss (1998), Algorithms for Separable Nonlinear Resource Allocation Problems, *Operations Research* 46, 272-284.

[7] Luss H., and S.K. Gupta (1975), Allocation of Effort Resources Among Competing Activities, *Operations Research* 23, 360-366.

[8] Nielsen S.S., and S.A. Zenios (1992), Massively Parallel Algorithms for Singly Constrained Convex Problems, *ORSA Journal on Computing* 4,no. 2, 166-181.

[9] Nielsen S.S., and S.A. Zenios (1993), A Massively Parallel Algorithm for Nonlinear Stochastic Network Problems, *Operations Research* 41, 319-337.

[10] Pardalos P.M., and N. Kovoor (1990), An Algorithm for a Single Constrained Class of Quadratic Programs Subject to Upper and Lower Bounds, *Mathematical Programming* 46, 321-328.

[11] Shetty B., and R. Muthukrishnan (1990), A Parallel Projection for the Multicommodity Network Model, *Journal of the Operational Research Society* 41, 837-842.

[12] Ventura J.A., and M.X. Weng (1995), Minimizing Single-Machine Completion-Time Variance, *Management Science* 41, 1448-1455.

[13] Zipkin P.H. (1980), Simple Ranking Methods for Allocation of One Resource, *Management Science* 26, 34-43.